We Claim:

1. A plug-in to a host, the plug-in providing one or more special capabilities to the host, the plug-in comprising:

core means for conducting typical operation of the plug-in by which the one or more special capabilities are carried out; and

interface means for interfacing between the core means and a calling entity with respect to operational status of the plug-in.

2. The plug-in of claim 1, wherein:

the interface means is a first interface means; and

the plug-in further comprises

second interface means for interfacing between the core means and the host with respect to the typical operation.

3. The plug-in of claim 1, wherein:

the host tends one or more storage devices and represents a part of a storage area network (SAN); and

the calling entity represents a storage manager of the SAN.

4. The plug-in of claim 1, wherein, relative to a client-server architecture, the following relationships apply:

the calling entity represents a client relative to the plug-in; and

the plug-in represents a server relative to the calling entity.

5. The plug-in of claim 4, wherein the client-server architecture is the JCore architecture such that the calling entity is a JCore client plug-in and the plug-in is a JCore server plug-in.

6. The plug-in of claim 1, further comprising:

butler means for gathering operational status information (Op_Stat_Info) representing the operational status of the plug-in.

7. The plug-in of claim 6, wherein the butler means is operable for performing the gathering of the Op_Stat_Info in an on-going manner while the plug-in is plugged-into the host.

8. The plug-in of claim 7, wherein the butler means is further operable to

cause one or more pieces of the Op_Stat_Info to be stored upon the one or more pieces being gathered initially, respectively; and

update the one or more pieces, respectively, as is appropriate relative to the on-going manner by which the butler means gathers the Op_Stat_Info.

9. The plug-in of claim 8, further comprising:

status data object (DO) means for storing the Op_Stat_Info;

wherein

the butler means is further operable for causing the Op_Stat_Info to be stored via the status DO means,

the status DO means is accessible by the calling entity via the interface means.

10. The plug-in of claim 6, wherein the butler means is operable for

initiating the gathering the Op_Stat_Info in response to receiving a request from the calling entity, and

for performing the gathering, once initiated, of the Op_Stat_Info in an ad hoc manner.

11. The plug-in of claim 10, further comprising:

status data object (DO) means for storing the Op_Stat_Info;

wherein the butler means is further operable for

causing the Op_Stat_Info to be stored via the status DO means, and

passing, upon completion of the gathering, the status DO to the calling entity.

12. A first plug-in to a host, the first plug-in providing one or more special capabilities to a host, the first plug-in comprising:

core means for conducting typical operation of the first plug-in by which the one or more special capabilities are carried out; and

interface means for interfacing between the core means and one or more second plug-ins loaded on one or more external, relative to the host, entities, respectively, regarding respective operational status of the one or more second plug-ins.

13. The first plug-in of claim 12, wherein:

the interface means is a first interface means; and

the first plug-in further comprises

second interface means for interfacing between the core means and the host with respect to the typical operation.

14. The first plug-in of claim 12, wherein:

the host is a first host;

the one or more external entities are one or more second hosts that tend one or more storage units, respectively;

the first host and the one or more second hosts represent a part of a storage area network (SAN); and

the host represents a storage manager of the SAN.

15. The first plug-in of claim 14, wherein, relative to a client-server architecture, the following relationships apply:

the first plug-in represents a client relative to the second plug-in; and

the second plug-in represents a server relative to the first plug-in.

16. The first plug-in of claim 15, wherein the client-server architecture is the JCore architecture such that the first plug-in is a JCore client plug-in and the second plug-in is a JCore server plug-in.

17. A method of operating a plug-in to a host, the plug-in providing one or more special capabilities to the host, the method comprising:

interfacing to make available, from the plug-in to an external calling entity relative to the host, operational status information (Op_Stat_Info) regarding the plug-in.

18. The method of claim 17, further comprising:

exchanging typical information between the plug-in and the host, the exchange of the typical information being a part of typical operation of the plug-in by which the one or more special capabilities are carried out.

19. The method of claim 17, wherein:

the host tends one or more storage devices and represents a part of a storage area network (SAN); and

the Op_Stat_Info is made available to a storage manager of the SAN.

20. The method of claim 17, further comprising:

gathering the Op_Stat_Info in an on-going manner while the plug-in is plugged-into the host.

21. The method of claim 20, further comprising:

storing one or more pieces of the Op_Stat_Info upon initially gathering the one or more pieces, respectively; and

updating the one or more pieces, respectively, as is appropriate relative to the on-going manner of the gathering step.

22. The method of claim 21, further comprising:

using a status data object (DO) to store the Op_Stat_Info; and

making the status DO accessible by the calling entity.

23. The method of claim 20, further comprising:

receiving a request from the calling entity for the Op_Stat_Info;

initiating the gathering step in response to receiving the request; and

performing the gathering step, once initiated, in an ad hoc manner.

24. The method of claim 23, further comprising:

using a status data object (DO) to store the Op_Stat_Info; and

passing, upon completing the gathering step, the status DO to the calling entity.

25. A machine-readable medium including instructions execution of which by a host produces a first plug-in, the first plug-in providing one or more special capabilities to the host, the machine-readable instructions comprising:

a core code segment for conducting typical operation of the first plug-in by which the one or more special capabilities are carried out; and

an interface code segment for interfacing between the core code segment and one or more second plug-ins loaded on one or more external, relative to the host, entities, respectively, regarding respective operational status of the one or more second plug-ins.

26. The machine-readable instructions of claim 25, wherein:

the interface code segment is a first interface code segment; and

the machine-readable instructions further comprise

a second interface code segment for interfacing between the core code segment and the host with respect to the typical operation.

27. The machine-readable instructions of claim 25, wherein:

the host is a first host; and

the one or more external entities are one or more second hosts that tend one or more storage units, respectively;

the first host and the one or more second hosts represent a part of a storage area network (SAN); and

the first host represents a storage manager of the SAN.

28. The machine-readable instructions of claim 25, wherein, relative to a client-server architecture, the following relationships apply:

the first plug-in represents a client relative to the second plug-in; and

the second plug-in represents a server relative to the first plug-in.

29. The machine-readable instructions of claim 28, wherein the client-server architecture is the JCore architecture such that the first plug-in is a JCore client plug-in and the second plug-in is a JCore server plug-in.

30. A method of operating a first plug-in to a host, the plug-in providing one or more special capabilities to a first host, the method comprising:

obtaining operational status information (Op_Stat_Info) from one or more second plug-ins loaded on one or more external, relative to the host, entities, respectively.

31. The method of claim 30, further comprising:

exchanging typical information between the first plug-in and the host, the exchange of the typical information being a part of typical operation of the plug-in by which the one or more special capabilities are carried out.

32. The method of claim 30, wherein:

the host is a first host; and

the one or more external entities are one or more second hosts that tend one or more storage units, respectively;

the first host and the one or more second hosts represent a part of a storage area network (SAN); and

the first host represents a storage manager of the SAN.

33. A machine-readable medium including instructions execution of which by a host produces a plug-in, the plug-in providing one or more special capabilities to the host, the machine readable instructions comprising:

a core code segment execution of which causes the one or more special capabilities to be carried out during typical operation of the plug-in; and

a status interface code segment for interfacing between the core and a calling entity and by which operational status of the plug-in is made available to the calling entity.

34. The machine-readable instructions of claim 33, further comprising:

a central interface code segment for interfacing between the core portion and the host and by which typical information is exchanged between the plug-in and the host, the exchange of the typical information being a part of the typical operation.

35. The machine-readable instructions of claim 33, wherein:

the host tends one or more storage devices and represents a part of a storage area network (SAN); and

the calling entity represents a storage manager of the SAN.

36. The machine-readable instructions of claim 33, wherein the plug-in, the host and the calling entity are configured according to the JCore architecture such that the calling entity represents a client relative to the plug-in and the plug-in represents a server relative to the calling entity.